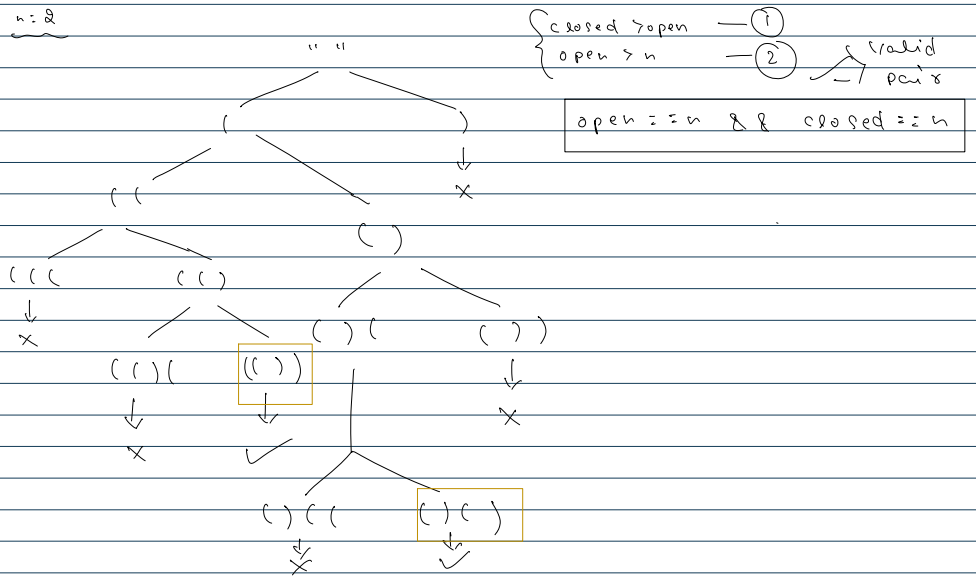
Recursion

n : 2 => ( ) ( ) (( ))

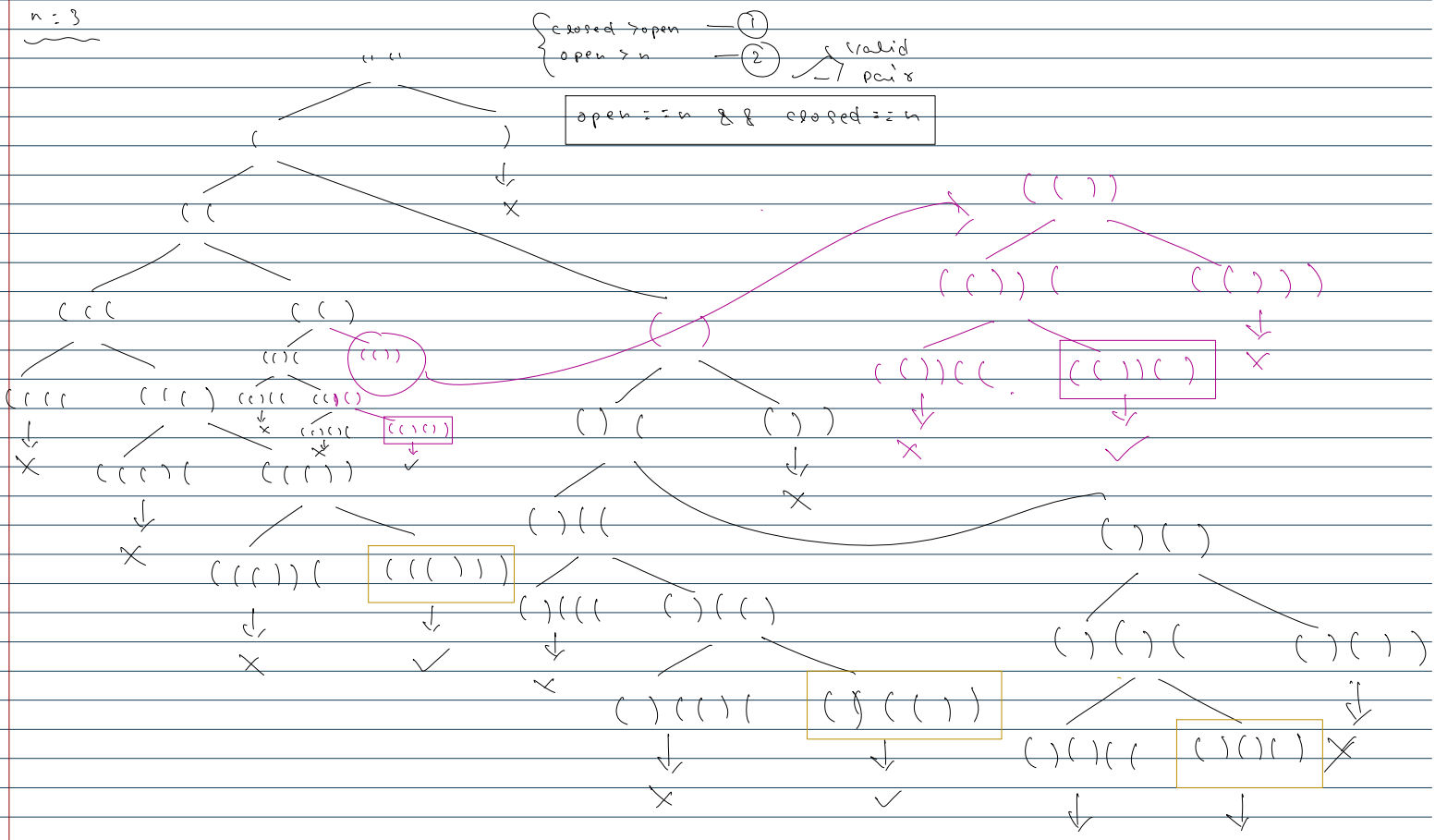
n : 3 => ((( ))) (( ) ( )) ( ) ( ) ( ) (( )) ( ) ( ) (( ))

options → (,) → choices → decision => Recursion

n : 2



n : 3

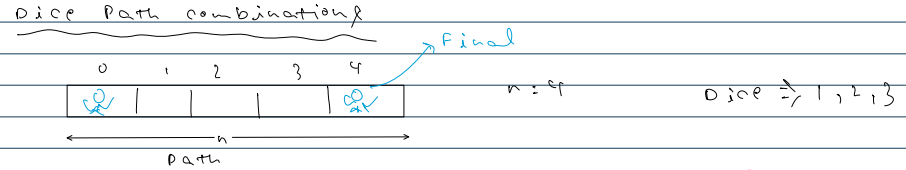


```
public class Main {
    public static void main(String[] args) {
        int n = 3;
        String ans = "";
        int open = 0;
        int closed = 0;
        paranthesis(n, open, closed, ans);
    }

    public static void paranthesis(int n, int open, int closed, String ans){
        if(open==n && closed==n){
            System.out.println(ans);
            return;
        }
        if(open > n || closed > open){
            return;
        }
        paranthesis(n, open+1, closed, ans+"(");
        paranthesis(n, open, closed+1, ans+")");
    }
}
```

```
class Solution {
    public List<String> generateParenthesis(int n) {

        List<String> opList = new ArrayList<>();
        String ans = "";
        int open = 0;
        int closed = 0;
        paranthesis(n, open, closed, ans, opList);

        return opList;
    }

    public static void paranthesis(int n, int open, int closed, String ans, List<String> opList){
        if(open==n && closed==n){
            opList.add(ans);
            // System.out.println(ans);
            return;
        }
        if(open > n || closed > open){
            return;
        }
        paranthesis(n, open+1, closed, ans+"(", opList);
        paranthesis(n, open, closed+1, ans+")", opList);
    }
}
```

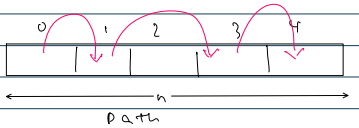
Board Path

Dice Path combinations



n = 4

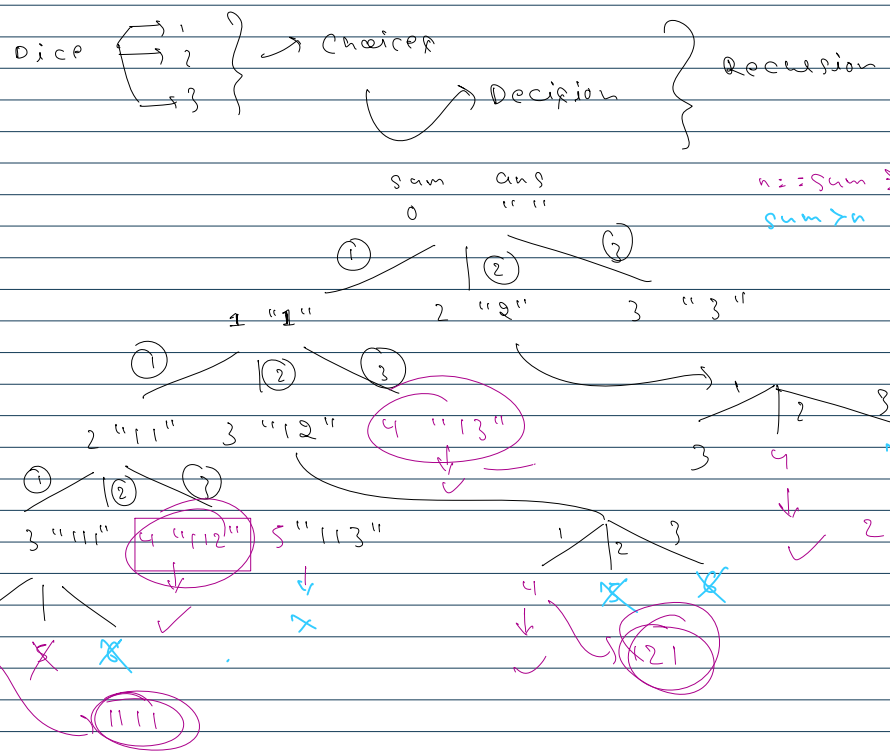
Dice => 1, 2, 3



1st time => 1
2nd time => 2
3rd time => 1

1 2 1
1 1 1 1
2 2
2 1 1
3 1

4




```
public class Main {
    public static void main(String[] args) {
        int n = 8;
        String ans = "";
        int sum = 0;
        printPath(n, sum, ans);
    }

    public static void printPath(int n, int sum, String ans){
        if(sum==n){
            System.out.println(ans);
            return;
        }
        if(sum>n){
            return;
        }

        for(int dice=1; dice<=6; dice++){
            printPath(n, sum+dice, ans+dice);
        }
        // printPath(n, sum+1, ans+1);
        // printPath(n, sum+2, ans+2);
        // printPath(n, sum+3, ans+3);
    }
}
```

Check character presence from index

str → "abcdefafghik"
index → 5
char → f

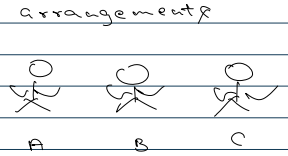
↳ whether char exists after index 5 or not.

```
public class Main {
    public static void main(String[] args) {
        String str = "fhgsfujasksalks";
        int index = 5;
        char ch = 's';
        System.out.println(isPresent(str, index, ch));
    }
```

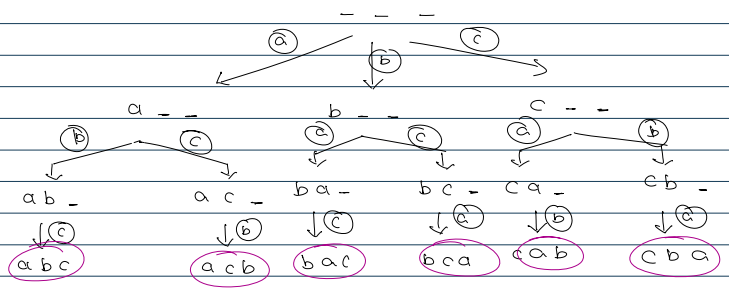
```
    public static boolean isPresent(String str, int index, char ch){
        for(int j=index; j<str.length(); j++){
            if(str.charAt(j) == ch){
                return true;
            }
        }
        return false;
    }
}
```

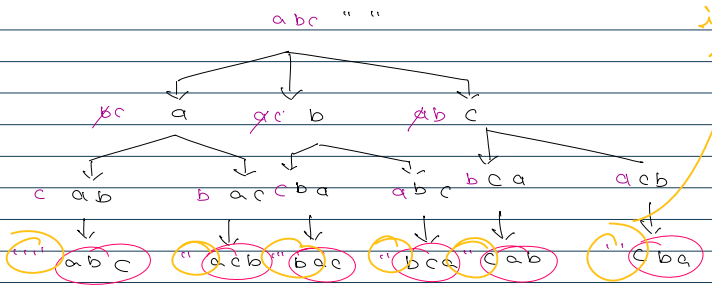
Permutations

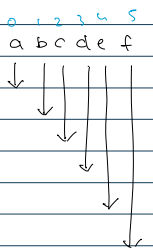
arrangements



a b c
a c b
b a c
b c a
c a b
c b a







i	ip(s1+s2)	op
0	- bcdef s(0,0) s(1)	op+'a'
1	a cdef s(0,1) s(2)	op+'b'
2	ab def s(0,2) s(3)	op+'c'
3	abc ef s(0,3) s(4)	op+'d'
4	abcd f s(0,4) s(5)	op+'e'
5	abcde - s(0,5) s(6)	op+'f'

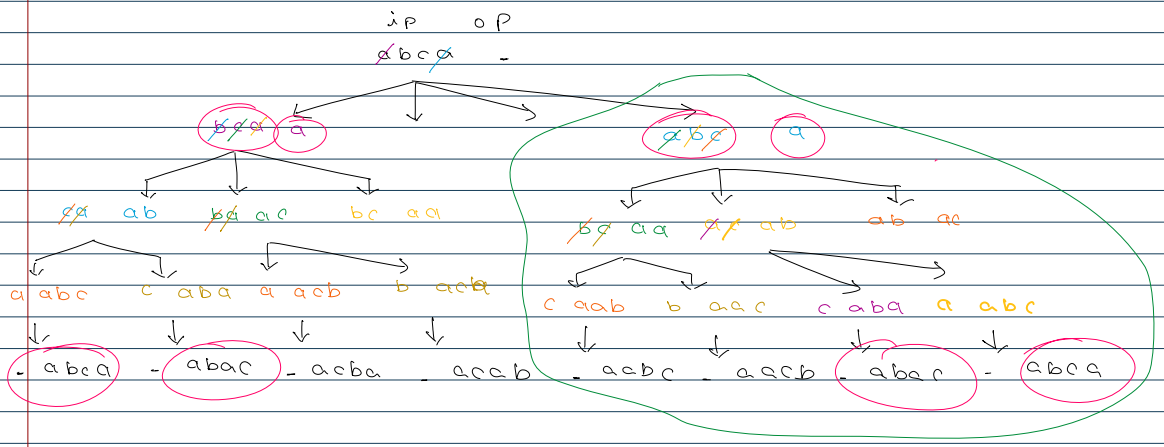
s1 = ip.substring(0,i)
s2 = ip.substring(i+1)

```
for(int i=0; i<ip.length(); i++){
    char ch = ip.charAt(i);
    s1 = ip.substring(0,i);
    s2 = ip.substring(i+1);
    permutation(s1+s2, op+ch);
}
```

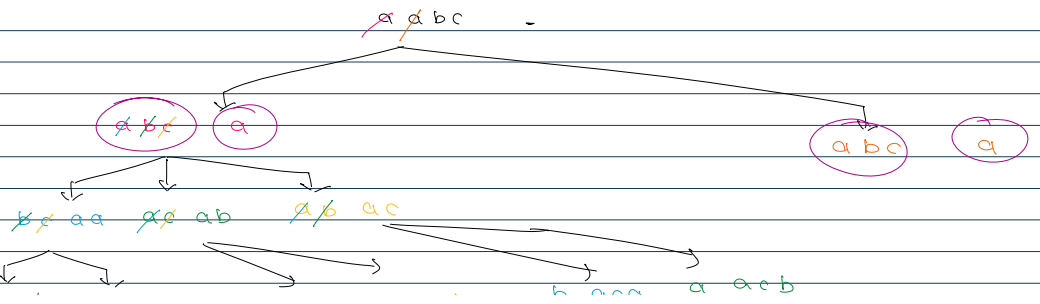
(ip)

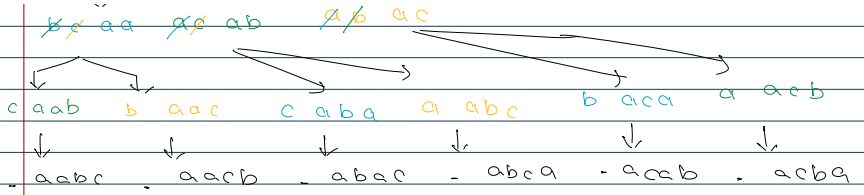
```
public class Main {
    public static void main(String[] args) {
        String ip = "abc";
        String op = "";
        permutation(ip, op);
    }

    public static void permutation(String ip, String op){
        if(ip.length()==0){
            System.out.println(op);
```

```
            return;
        }
        for(int i=0; i<ip.length(); i++){
            char ch = ip.charAt(i);
            String s1 = ip.substring(0, i);
            String s2 = ip.substring(i+1);

            permutation(s1+s2, op+ch);
        }
    }
}
```



```
public class Main {
    public static void main(String[] args) {
        String ip = "abca";
        String op = "";
        permutation(ip, op);
    }

    public static void permutation(String ip, String op){
        if(ip.length()==0){
            System.out.println(op);
            return;
        }
        for(int i=0; i<ip.length(); i++){
            char ch = ip.charAt(i);
            if(isPresent(ip, i+1, ch ) == false){
                String s1 = ip.substring(0, i);
                String s2 = ip.substring(i+1);

                permutation(s1+s2, op+ch);
            }
        }
    }

    public static boolean isPresent(String str, int index, char ch){
        for(int j=index; j<str.length(); j++){
            if(str.charAt(j) == ch){
                return true;
            }
        }
        return false;
    }

}
```




aa
ab
ac
c aab
b aac
c aba
a abc
b aca
a acb
aabc
aacb
abac
abca
acab
acba